**SDLC MODELS**

**Popular SDLC Models**

This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases. Here, are some most important phases of SDLC life cycle:

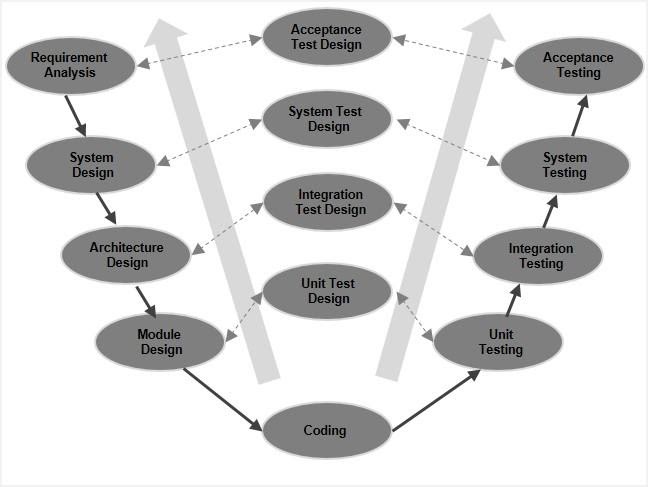
1. **V-Model**

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

**V-Model - Design**

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the ‘V’ and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

**The following illustration depicts the different phases in a V-Model of the SDLC.**



**There are several Verification phases in the V-Model, each of these are explained in detail below.**

**Business Requirement Analysis**

This is the first phase in the development cycle where the product requirements are understood from the customer’s perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The **acceptance test design planning** is done at this stage as business requirements can be used as an input for acceptance testing.

**System Design**

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

**Architectural Design**

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD).**

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

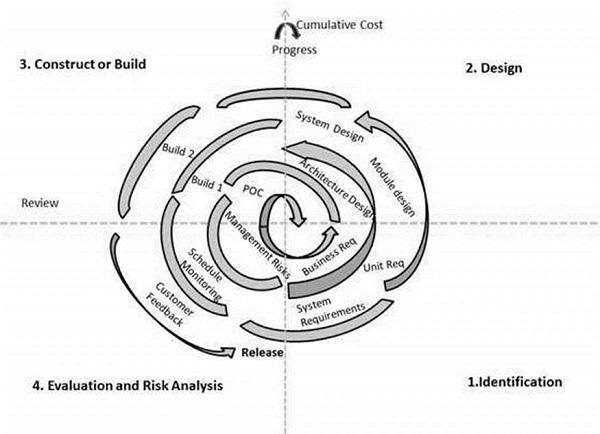
**Module Design**

In this phase, the detailed internal design for all the system modules is specified, referred to as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

**2. Spiral Model**

The spiral model is a risk-driven process model. This SDLC model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. This model adopts the best features of the prototyping model and the waterfall model. The spiral methodology is a combination of rapid prototyping and concurrency in design and development activities.

**The following illustration is a representation of the Spiral Model, listing the activities in each phase.**



The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

**Identification**

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase. This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

**Design**

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

**Construct or Build**

The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback. Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

**Evaluation and Risk Analysis**

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

**3. Big bang model**

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come. This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

**Big Bang Model - Pros and Cons**

The advantage of this Big Bang Model is that it is very simple and requires very little or no planning. Easy to manage and no formal procedure are required. However, the Big Bang Model is a very high-risk model and changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project. It is ideal for repetitive or small projects with minimum risks.

**The advantages of the Big Bang Model are as follows:**

* This is a very simple model
* Little or no planning required
* Easy to manage
* Very few resources required
* Gives flexibility to developers
* It is a good learning aid for new comers or students.

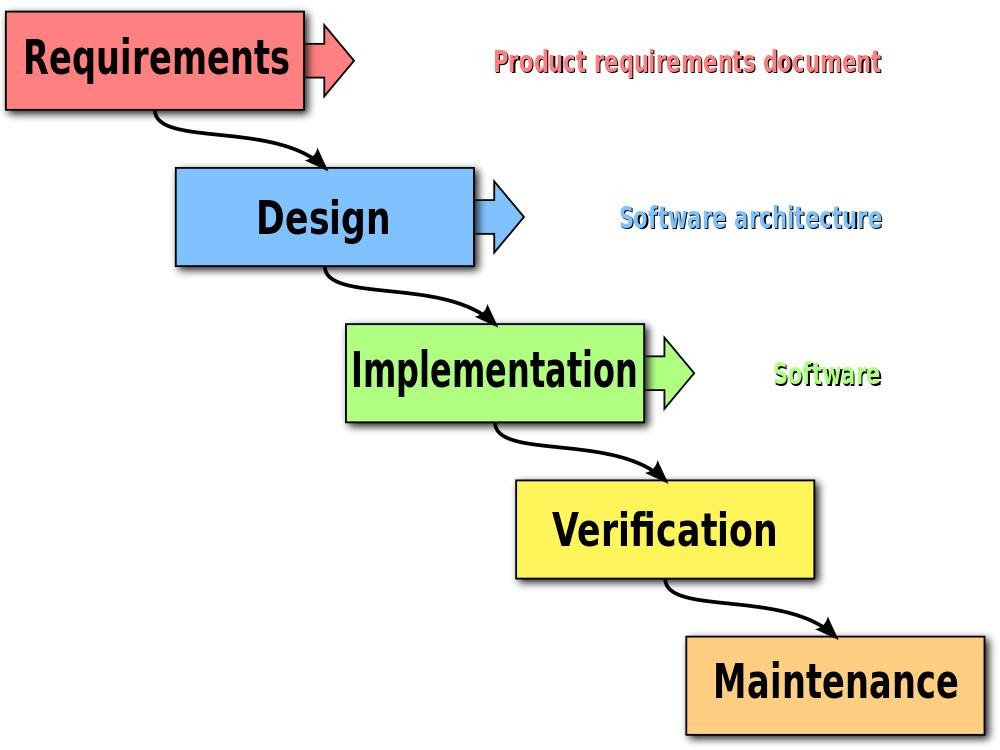
**The disadvantages of the Big Bang Model are as follows:**

* Very High risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Can turn out to be very expensive if requirements are misunderstood.

**4. Waterfall model**

The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases. In this SDLC model, the outcome of one phase acts as the input for the next phase.

Waterfall methodology begins with long planning and design phases. Once developed, the software then goes through phases of testing, and is finally deployed for use. Waterfall is considered by many to be too rigid to adapt to changing requirements. It does not support feedback throughout the process, leading to the implementation of requirements that may have changed during the development effort. This weakness in Waterfall led to the development of more flexible methodologies, such as Agile.



The Waterfall method of software development follows a rigid, predetermined path through a set of phases. This method was adapted from traditional engineering. Ironically, the paper credited as the origin of the Waterfall method, describes it as being fundamentally flawed. The method that is known today as “Waterfall” was mistakenly derived from a misunderstanding of this original work. Despite that fact, Waterfall became a very common, even standard methodology for large projects around the world.

**Waterfall Model - Advantages**

* The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.
* Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.
* Some of the major advantages of the Waterfall Model are as follows −
* Simple and easy to understand and use.
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented.